



The Ultimate Developers Toolkit

Jonathan Zarge
Dan Ginsburg

February 20, 2008

Agenda

GPU PerfStudio

GPU ShaderAnalyzer

RenderMonkey

Additional Tools

- Tootle
- GPU MeshMapper
- CubeMapGen
- The Compressorator
- OpenGL ES 2.0 Emulator

All tools available today at:
<http://ati.amd.com/developer/>



GPU PerfStudio

GPU PerfStudio – Why use it?

- Is the graphics performance of your game not meeting expectations?
- Are you wondering what the heck the graphics hardware is doing?

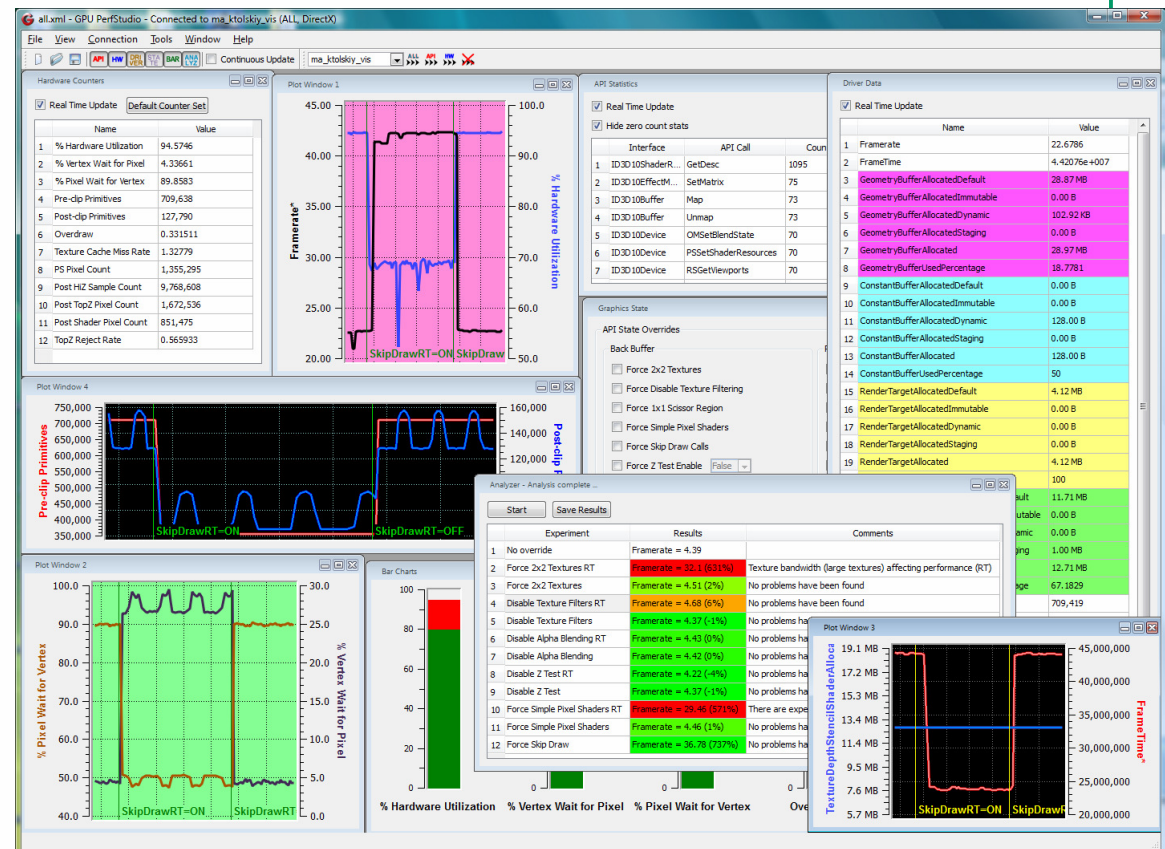
The solution is GPU PerfStudio:

- Interactive tool for enabling graphics performance optimization
- View into API, driver and hardware
- Current being used by:
 - Leading game developers
 - AMD graphics ISV Engineering team
 - AMD demo and driver teams



GPU PerfStudio

- Performance analysis and optimization tool
- Real-time visualization:
 - API statistics
 - Hardware/driver data
- Override rendering states
- Launch remote applications
- Flexible data visualization
- Bottleneck analysis



GPU PerfStudio – New Features

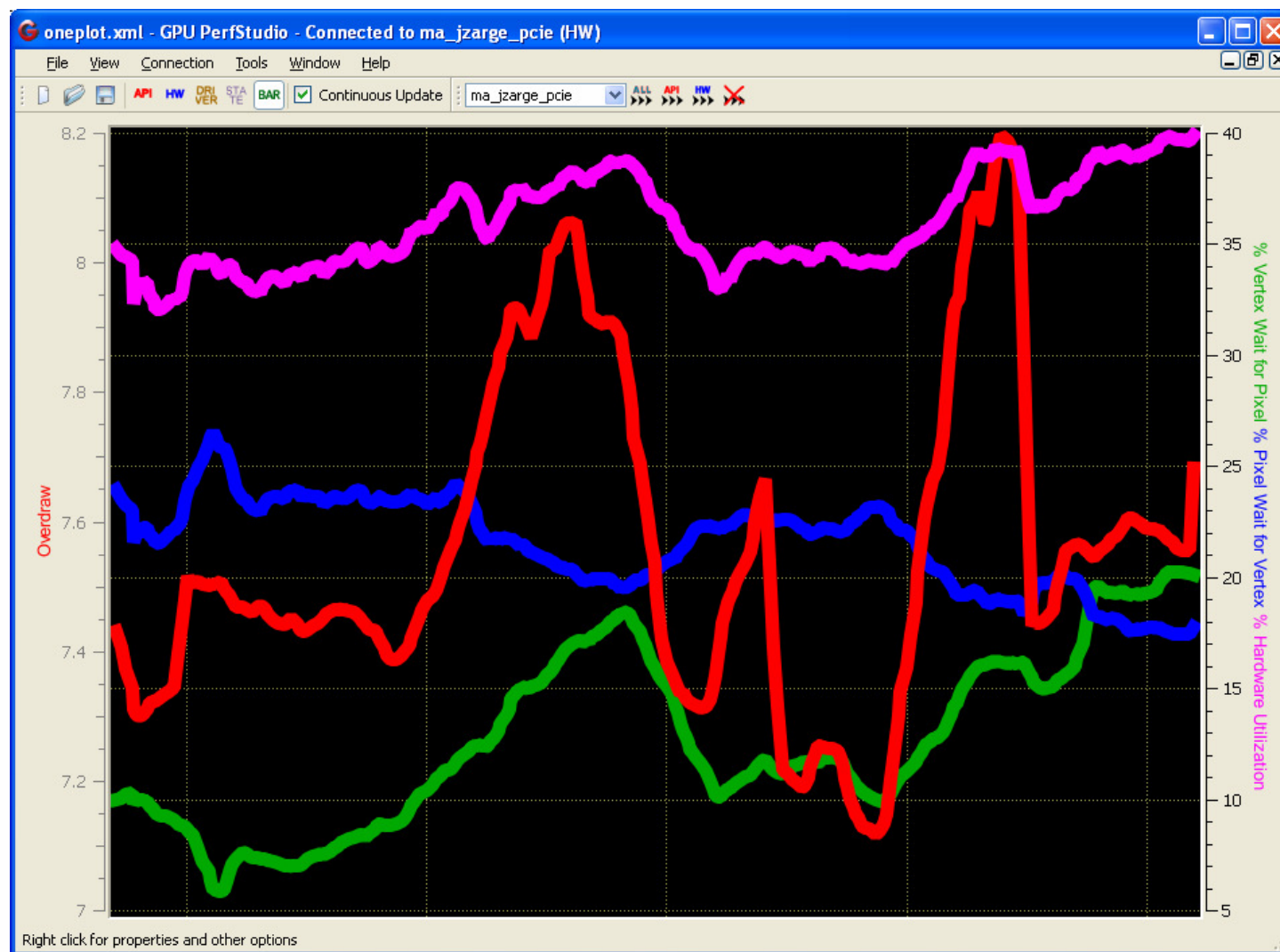
- D3D10 and OpenGL support
- ATI Radeon HD2000 and HD3000 support
- Automated bottleneck analysis:

Analyzer - Analysis complete ...

Start Save Results

	Experiment	Results	Comments
1	No override	Framerate = 4.39	
2	Force 2x2 Textures RT	Framerate = 32.1 (631%)	Texture bandwidth (large textures) affecting performance (RT)
3	Force 2x2 Textures	Framerate = 4.51 (2%)	No problems have been found
4	Disable Texture Filters RT	Framerate = 4.68 (6%)	No problems have been found
5	Disable Texture Filters	Framerate = 4.37 (-1%)	No problems have been found
6	Disable Alpha Blending RT	Framerate = 4.43 (0%)	No problems have been found
7	Disable Alpha Blending	Framerate = 4.42 (0%)	No problems have been found
8	Disable Z Test RT	Framerate = 4.22 (-4%)	No problems have been found
9	Disable Z Test	Framerate = 4.37 (-1%)	No problems have been found
10	Force Simple Pixel Shaders RT	Framerate = 29.46 (571%)	There are expensive pixel shaders (RT)
11	Force Simple Pixel Shaders	Framerate = 4.46 (1%)	No problems have been found
12	Force Skip Draw	Framerate = 36.78 (737%)	No problems have been found

GPU PerfStudio - Demo



<http://ati.amd.com/developer/tools.html>

GPU ShaderAnalyzer

GPU ShaderAnalyzer

Shader performance analysis tool:

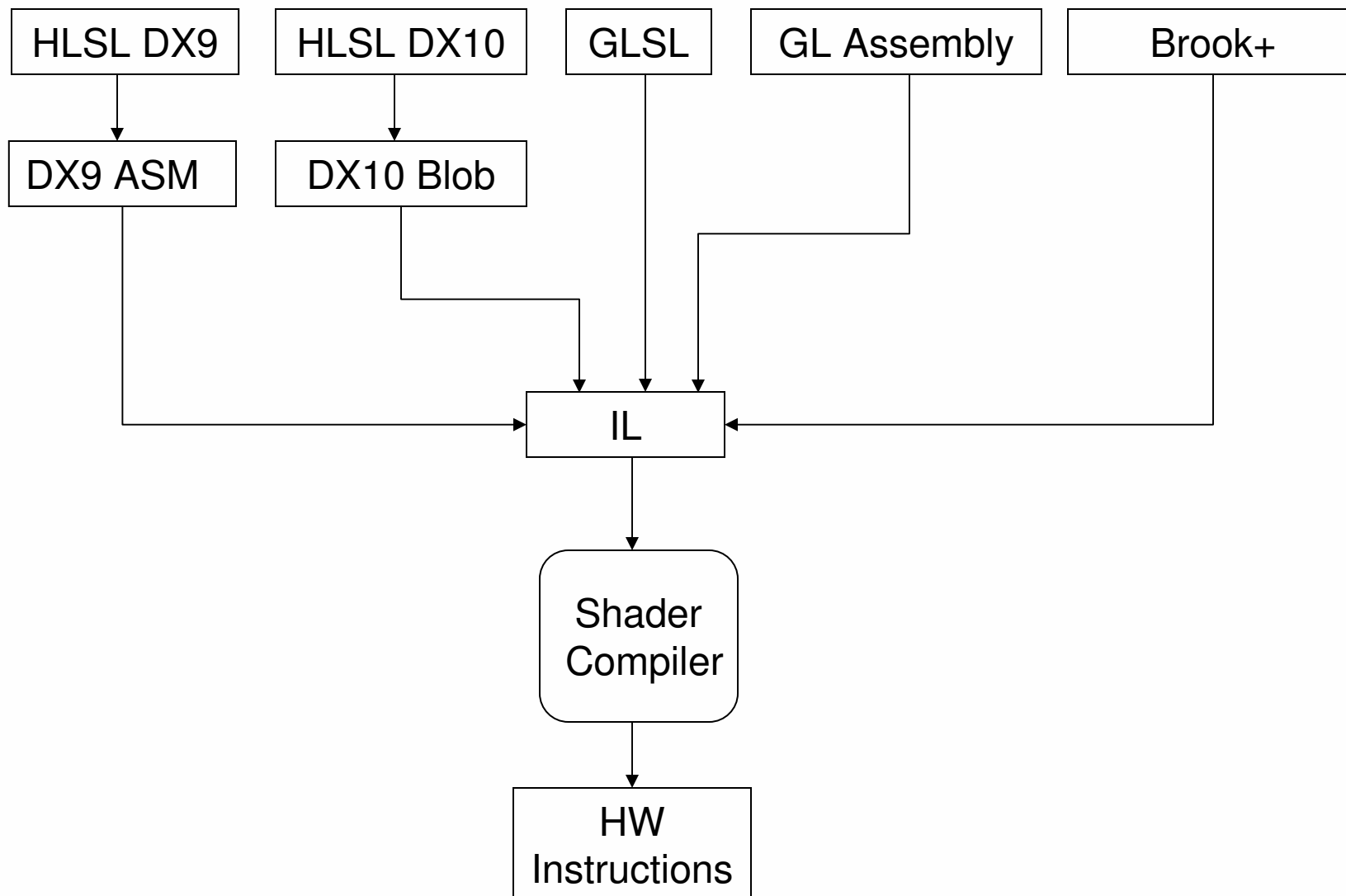
- View hardware disassembly
- Instant feedback as you tune
- Estimated cycle counts for all AMD GPUs

Support for all shading languages:

- HLSL, DX assembly
- DX9, DX10, DX10.1
- GLSL, ARB assembly
- Brook+, IL
- Vertex, fragment, and geometry shaders



GPU Shader Compilation



GPU ShaderAnalyzer - Demo

GPU ShaderAnalyzer - MetallicFlakes.fx

File Edit Help

Source Code

```
222 // volume noise
223 float4 Noise = tex3D(SparkleNoise, In.NoiseCoord);
224 // noisy diffuse of metal
225 Diffuse = In.Diffuse * Noise.a;
226 // glossy specular of wax
227 Specular = In.Specular;
228 Specular *= Specular;
229 Specular *= Specular;
230 // glossy reflection of wax
231 Gloss = texCUBE(Environment, In.Reflection) *
232 saturate(In.Glossiness);
233 // specular sparkle of flakes
234 Sparkle = saturate(dot((saturate(In.HalfVector) - 0.5)
235 (Noise.rgb - 0.5) * 2));
236 Sparkle *= Sparkle;
237 Sparkle *= Sparkle;
238 Sparkle *= Sparkle;
239 Sparkle *= Sparkle;
240 Sparkle *= k_s;
241 // combine the contributions
242 Color.rgb = Diffuse + Specular + Gloss + Sparkle;
243 Color.w = 1;
```

HLSL Compiler

Compile Options

Target: ps_3_0

☐ Avoid Flow Control ☐ Prefer Flow Control

☐ Skip Optimization

Macro Definitions

Symbol Value

Right-click to add macros.

Object Code

Format: Radeon x1900 (R580) Assembly

US Program:

```
0 tex 00 : r03.rgba = lookup(r03.rgbr, tex01) ign_unc
1 tex 01 : r02.rgb_ = lookup(r02.rgbr, tex00) sem_wait sem_grab
2 alu 00 rgb: r05.rgb = clamped mad(r05.rgb, 1.0, 0.0)
alpha: r00.a = mad(r01.b, r01.b, 0.0)
3 alu 01 rgb: r05.rgb = mad(r05.rgb, 1.0, c01.rrr)*2
alpha: r00.a = mad(r00.a, r00.a, 0.0)
4 alu 02 rgb: r03.rgb = mad(r03.rgb, 1.0, c01.rrr)*2 se
alpha: r02.a = mad(r00.b, r03.a, r00.a)
5 alu 03 rgb: r03.r-- = clamped dp3(r05.rgb, r03.rgb)
mad(0.0, 0.0, 0.0)
alpha: r03.r-- = mad(r03.r, r03.r, 0.0)
6 alu 04 rgb: r03.-gb = mad(r01.0.0rg, r01.0.0rg, 0.0)
alpha: r04.a = mad(r03.r, r03.r, 0.0)
7 alu 05 rgb: r03.-gb = mad(r03.0.0gb, r03.0.0gb, 0.0)
alpha: r04.a = mad(r04.a, r04.a, 0.0)
8 alu 06 rgb: r00.-gb = mad(r00.0.0rg, r03.raa, r03.0.0
alpha: r04.a = mad(r04.a, r04.a, 0.0)
9 alu 07 rgb: r01.rgb = clamped mad(r04.rgb, 1.0, 0.0)
alpha: r01.a = mad(r04.a, r04.a, 0.0)
10 alu 08 rgb: r00.-gb = mad(r02.0.0rg, r01.0.0rg, r00.
alpha: r00.a = mad(r02.b, r01.b, r02.a)
11 alu 09 rgb: out0.rgb = mad(r01.aaa, c00.rgb, r00.gba)
alpha: out0.a = mad(1.0, 1.0, 0.0) last
```

Compiler Statistics (Using GPUShaderAnalyzer)

ASIC	Registers	Cycles (Bilinear)	Cycles (Trilinear)	Cycles (Aniso)	ALU:Tex (Bilinear)	ALU:Tex (Trilinear)	ALU:Tex (Aniso)
Radeon 9700 (R300)	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Radeon x800 (R420)	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Radeon x850 (R480)	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Radeon x1800 (R520)	6	10.00	10.00	10.00	5.00	4.17	3.57
Radeon x1900 (R580)	6	4.00	4.00	4.00	2.00	1.67	1.43

D3D Assembly Statistics

Shader Version = 3.0

Instruction Count = 28

ALU Instructions = 17, Texture Instructions = 2, ALU:Texture Ratio = 8.50

Constant Register Count = 2

Temp Register Count = 3, Sampler Register Count = 2, Input Register Count = 6, Output Register Count = 1

Has PS2.0 Instructions

Uses Arbitrary Swizzle

RenderMonkey

RenderMonkey

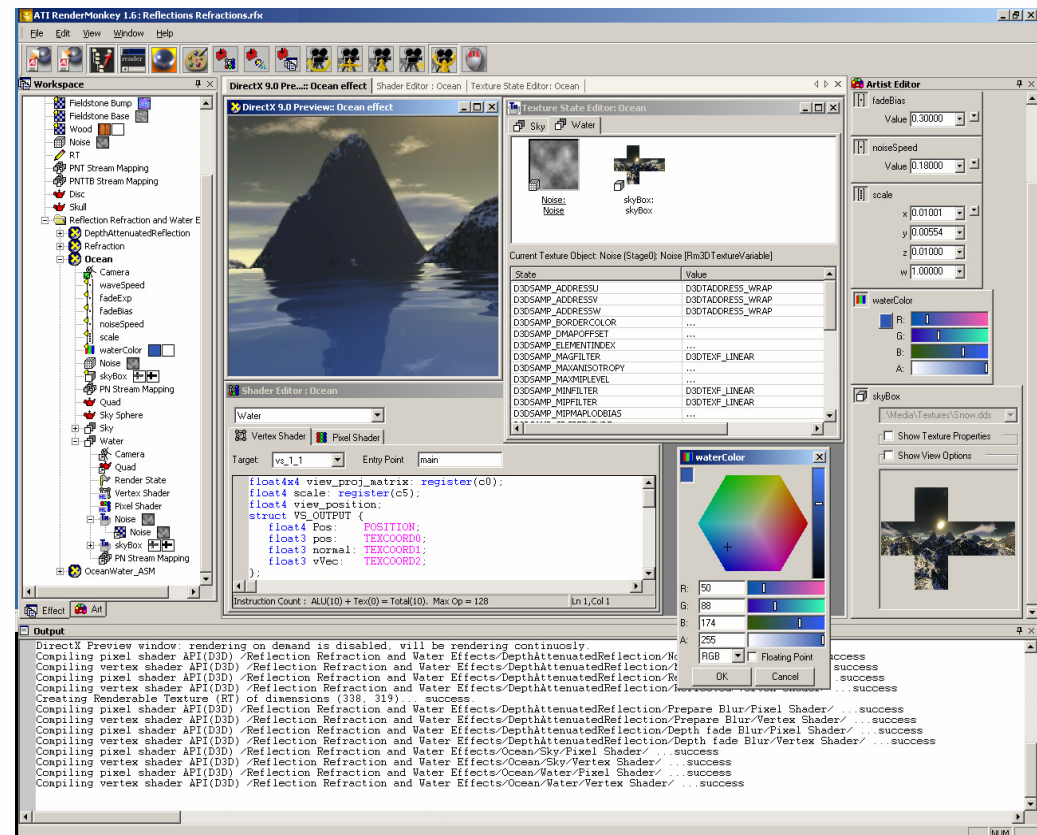


Shader Development Environment

- Rapid Prototyping of Shader Effects

Multiple Shading Languages

- DirectX HLSL
- DirectX Assembler
- OpenGL Shading Language
- OpenGL ES Shading Language



RenderMonkey – Why use it?

Full IDE for shader effect development

- Programmer and artist view for rapid iteration

Easy integration into game pipeline

- Plug-in SDK for custom import/export
 - Effects, models, textures, variables, etc.
- Support for many standard formats
 - DDS, BMP, TGA, X, OBJ, 3DS, DAE, FX, COLLADA FX

Encompasses all effect resources

- Render state, texture state, variables, render targets, textures, models, etc...



RenderMonkey – What's new?

V1.7:

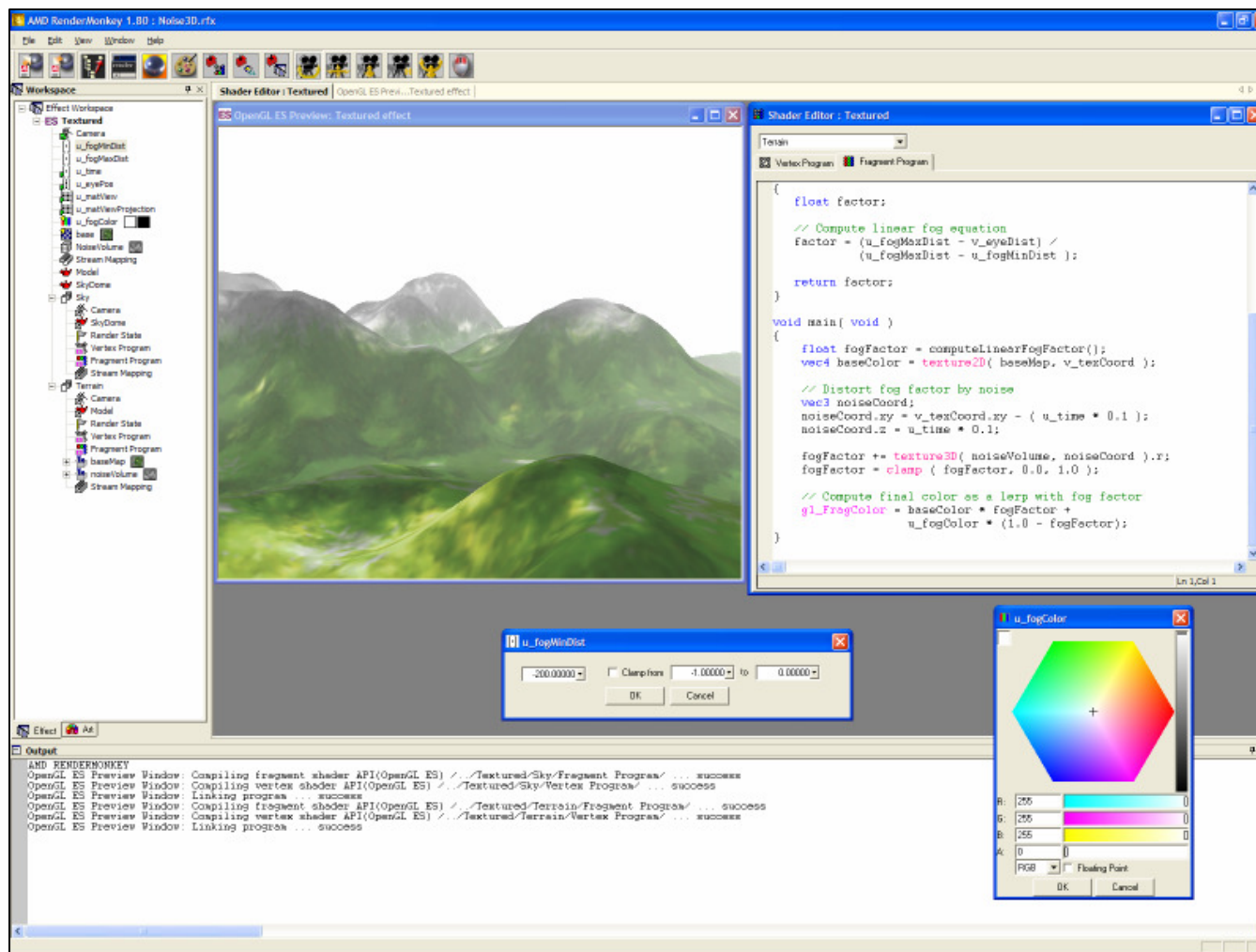
- Support for OpenGL ES 2.0
- ES Shading Language v1.00
- ES syntax highlighting
- ES render/sampler states
- Large suite of ES examples

V1.8:

- OpenGL COLLADA Effects Exporter
- Import COLLADA Geometry



RenderMonkey - Demo



Additional Tools

<http://ati.amd.com/developer/tools.html>

Tootle

- A Triangle Order Optimization Tool
- Provided as a library for integration into your tool-chain
- Improves vertex cache hit rate
 - Shade fewer vertices
- Reduces overdraw
 - Shade fewer pixels
 - View independent



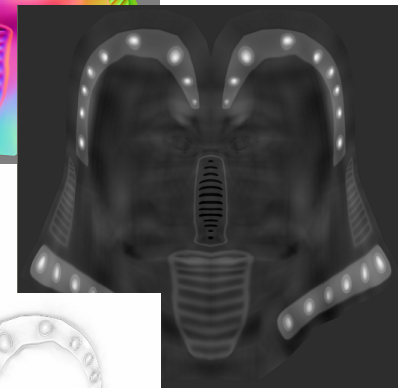
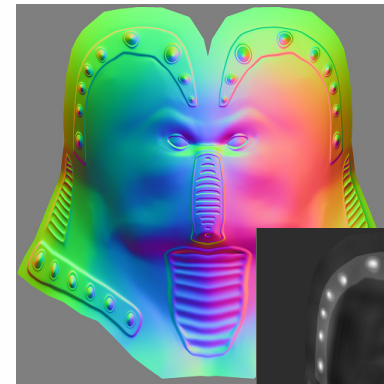
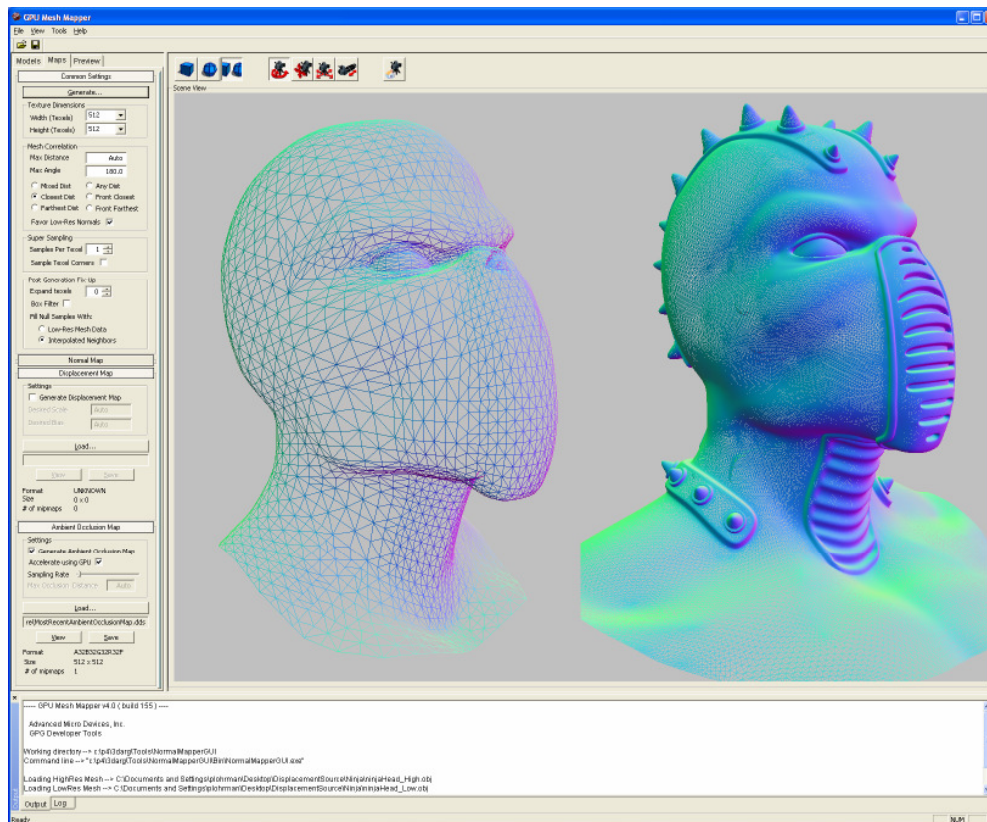
Tootle: Overdraw Reduction

- Example Scene
 - 70k polygons
 - 10 materials
- Reduced overdraw by factor of two
- 3-7% performance increase compared to D3DXOptimizeMesh.



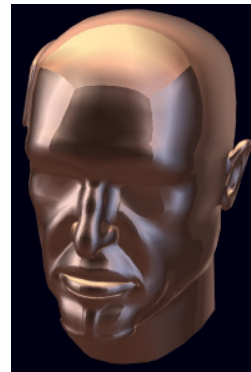
GPU MeshMapper

Use low and high detail models to generate normal maps, displacement maps, and ambient occlusion maps



CubeMapGen

- Standard mip-maps
 - Filters individual faces
 - Results in artifacts



- CubMapGen mip-maps
 - Pre-filters across faces
 - Removes visible edge

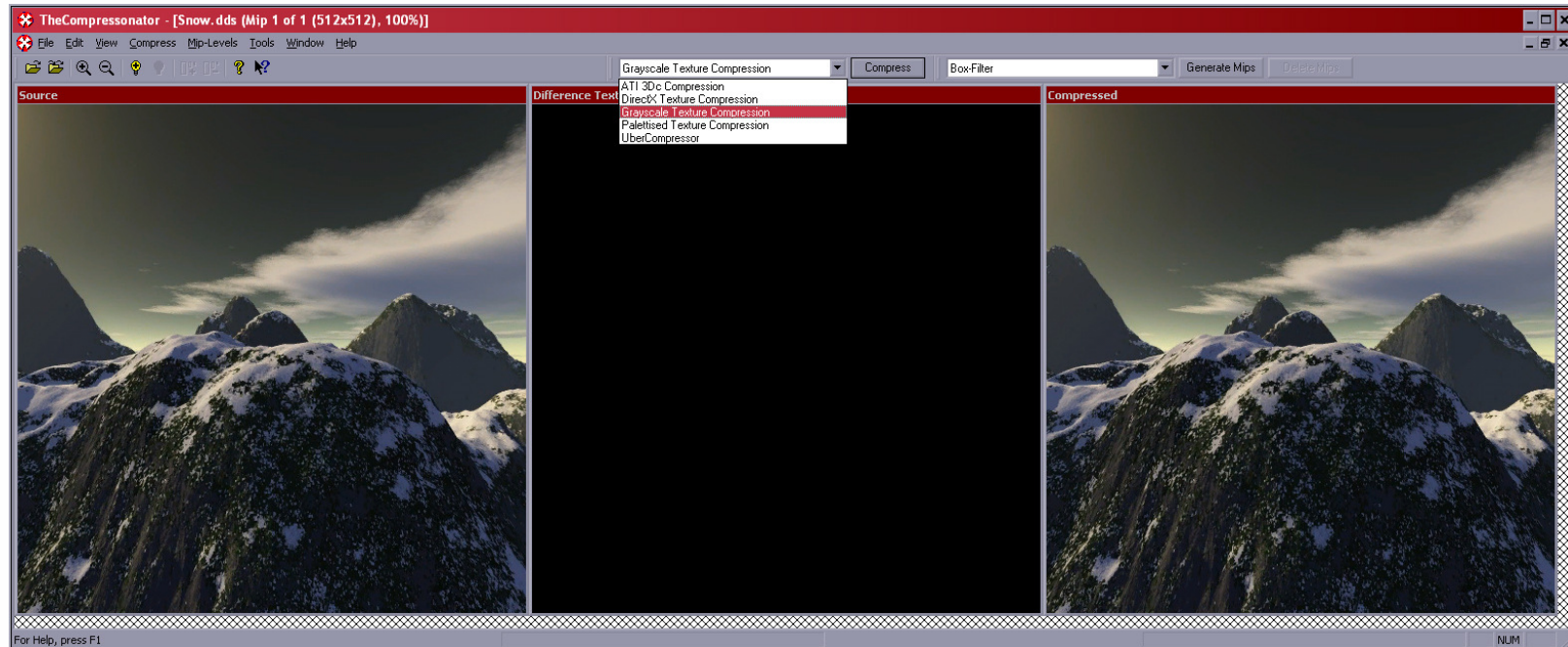


- Smaller, better cube maps at no rendering cost!

The Compressorator

Compresses textures to all AMD-supported formats

Visual diff to assess compression quality



OpenGL ES 2.0 Emulator

- Provides an OpenGL ES 2.0 development environment on the PC
 - Develop your OpenGL ES 2.0 applications today
 - Minimizes porting effort once hardware is available
- Full implementation of OpenGL ES 2.0 and EGL 1.3



Summary

GPU PerfStudio

GPU ShaderAnalyzer

RenderMonkey

Additional Tools

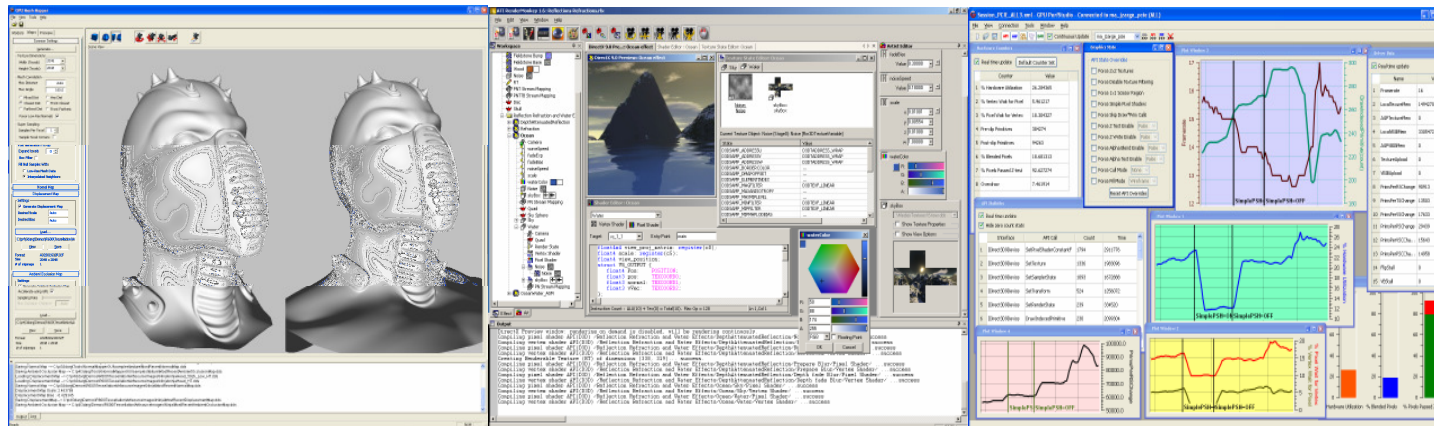
- Tootle
- GPU MeshMapper
- CubeMapGen
- The Compressorator
- OpenGL ES 2.0 Emulator



AMD at GDC 2008

Wednesday, February 20th Room 2015, West Hall	
9:00 – 10:00 ^{AM}	The Ultimate Developers Toolkit: Jonathan Zarge, Dan Ginsburg
10:30 – 11:30 ^{AM}	Harnessing the Power of Multiple GPUs: Jon Story and Holger Gruen
12:00 – 1:00 ^{PM}	Ultimate Graphics Performance for DirectX10 Hardware: Nicolas Thibieroz
2:30 – 3:30 ^{PM}	Optimization Techniques for Attacking CPU Data Bottlenecks in PC Games: Michael Wall
4:00 – 5:00 ^{PM}	Tessellation in a Low Poly World: Bill Bilodeau and Peter Lohrmann

Questions?



Jonathan.Zarge@amd.com

Dan.Ginsburg@amd.com

GPUTools.Support@amd.com

<http://ati.amd.com/developer/>

<http://ati.amd.com/developer/tools.html>